Python Modules and Packages

from https://docs.python.org/3/tutorial/modules.html

Python Modules

- Module: A file containing Python definitions and statements
- The file name (without the .py suffix) is the module name
- The module name is the global namespace when using the module functions/variables
- Now the code can be used with import everywhere in the same computer

```
def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()
```

```
>>> import fibo
```

```
>>> fibo.fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

```
>>> fib = fibo.fib
>>> fib(500)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

Python Modules

- A way to lift the module elements to the global space

```
>>> from fibo import fib
>>> fib(500)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

```
>>> from fibo import *
>>> fib(500)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

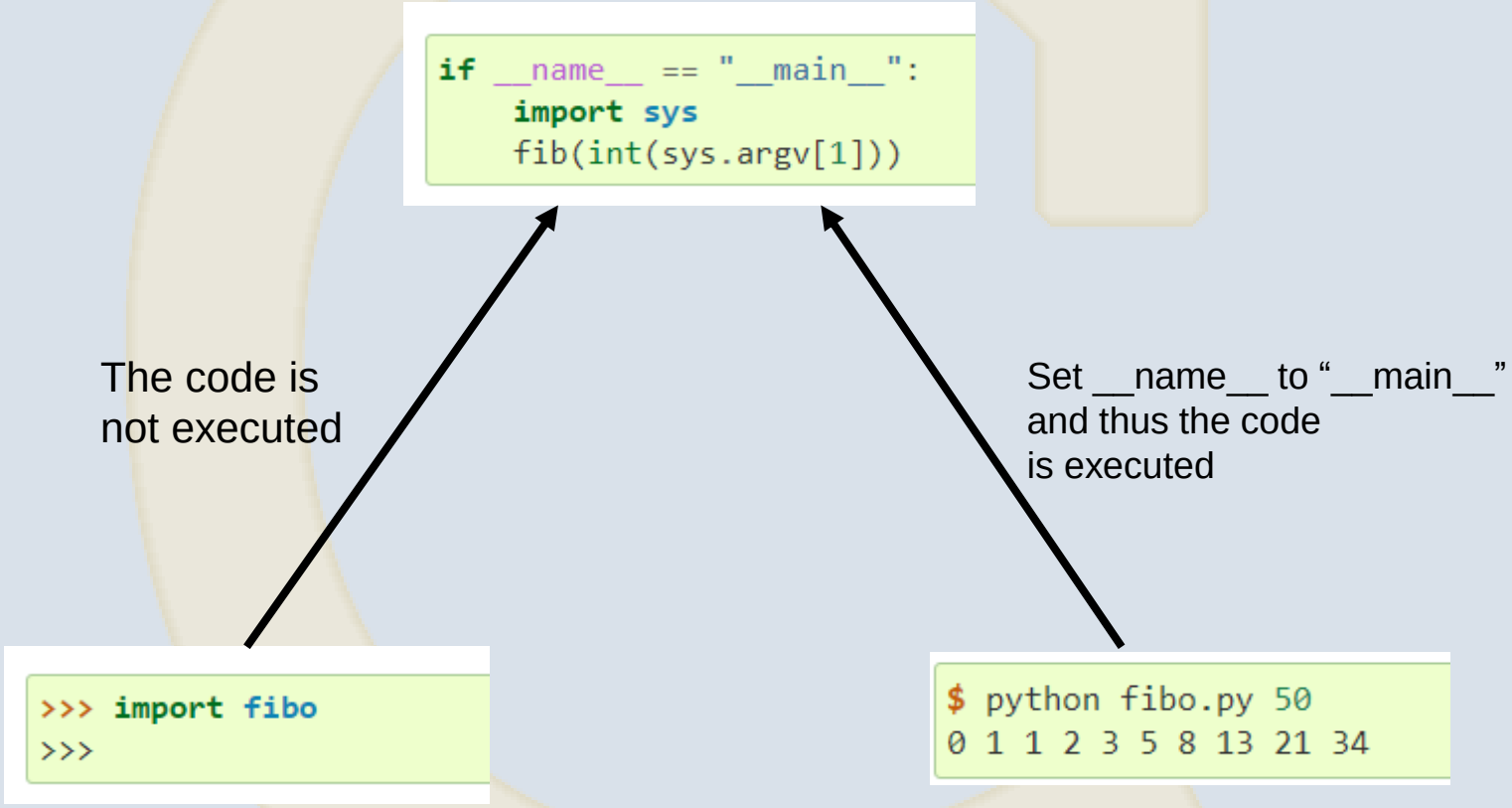
Risky as the global space is contaminated

- Renaming the module

```
>>> import fibo as fib
>>> fib.fib(500)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

```
>>> from fibo import fib as fibonacci
>>> fibonacci(500)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

Executing Modules



Module Search

- When importing a module, it is searched in this order

1. Searching a built-in module with this name
2. Search in the directories given by the global variable sys.path

sys.path is initialized with:

1. The directory containing the input script
2. The directories listed at the environment variable PYTHONPATH
3. The installation-dependent default

Updating sys.path:

```
>>> import sys
>>> sys.path.append('/ufs/guido/lib/python')
```

Standard Modules

- Python comes with a library of standard modules
- Not part of the language, but very useful and comfortable to have
  - Math functionality
  - System calls
  - Etc.
- These are also operating system and platform dependent

Dir()

The built-in function `dir()` is used to find out which names a module defines. It returns a sorted list of strings:

```
>>> import fibo, sys
>>> dir(fibo)
['__name__', 'fib', 'fib2']
>>> dir(sys)
['__breakpointhook__', '__displayhook__', '__doc__', '__excepthook__',
 '__interactivehook__', '__loader__', '__name__', '__package__', '__spec__',
 '__stderr__', '__stdin__', '__stdout__', '__unraisablehook__',
 '_clear_type_cache', '_current_frames', '_debugmallocstats', '_framework',
 '_getframe', '_git', '_home', '_xoptions', 'abiflags', 'addaudithook',
 'api_version', 'argv', 'audit', 'base_exec_prefix', 'base_prefix',
 'breakpointhook', 'builtin_module_names', 'byteorder', 'call_tracing',
 'callstats', 'copyright', 'displayhook', 'dont_write_bytecode', 'exc_info',
 'excepthook', 'exec_prefix', 'executable', 'exit', 'flags', 'float_info',
 'float_repr_style', 'get_asyncgen_hooks', 'get_coroutine_origin_tracking_depth',
 'getallocatedblocks', 'getdefaultencoding', 'getdlopenflags',
 'getfilesystemencodeerrors', 'getfilesystemencoding', 'getprofile',
 'getrecursionlimit', 'getrefcount', 'getsizeof', 'getswitchinterval',
 'gettrace', 'hash_info', 'hexversion', 'implementation', 'int_info',
 'intern', 'is_finalizing', 'last_traceback', 'last_type', 'last_value',
 'maxsize', 'maxunicode', 'meta_path', 'modules', 'path', 'path_hooks',
 'path_importer_cache', 'platform', 'prefix', 'ps1', 'ps2', 'pycache_prefix',
 'set_asyncgen_hooks', 'set_coroutine_origin_tracking_depth', 'setdlopenflags',
 'setprofile', 'setrecursionlimit', 'setswitchinterval', 'settrace', 'stderr',
 'stdin', 'stdout', 'thread_info', 'unraisablehook', 'version', 'version_info',
 'warnoptions']
```

Without arguments, `dir()` lists the names you have defined currently:

```
>>> a = [1, 2, 3, 4, 5]
>>> import fibo
>>> fib = fibo.fib
>>> dir()
['__builtins__', '__name__', 'a', 'fib', 'fibo', 'sys']
```

variables,
modules,
functions, etc.

Built-in Functions

```
>>> import builtins
>>> dir(builtins)
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException',
 'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning',
 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError',
 'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarning',
 'EOFError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False',
 'FileExistsError', 'FileNotFoundError', 'FloatingPointError',
 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError',
 'ImportWarning', 'IndentationError', 'IndexError', 'InterruptedError',
 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError',
 'MemoryError', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented',
 'NotImplementedError', 'OSError', 'OverflowError',
 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError',
 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning',
 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError',
 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError',
 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError',
 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning',
 'ValueError', 'Warning', 'ZeroDivisionError', '_', '__build_class__',
 '__debug__', '__doc__', '__import__', '__name__', '__package__', 'abs',
 'all', 'any', 'ascii', 'bin', 'bool', 'bytearray', 'bytes', 'callable',
 'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credits',
 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval', 'exec', 'exit',
 'filter', 'float', 'format', 'frozenset', 'getattr', 'globals', 'hasattr',
 'hash', 'help', 'hex', 'id', 'input', 'int', 'isinstance', 'issubclass',
 'iter', 'len', 'license', 'list', 'locals', 'map', 'max', 'memoryview',
 'min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property',
 'quit', 'range', 'repr', 'reversed', 'round', 'set', 'setattr', 'slice',
 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple', 'type', 'vars',
 'zip']
```

Packages

- Collections of modules arranged in directories

Serves as a notifier that it is a package with subdirectories. Usually empty but can conain initialization code such as global definitions or imports

```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/                  Subpackage for sound effects
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/                  Subpackage for filters
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
```

Packages

- Using the packages
  - Importing sub-packages or modules

```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
```

```
import sound.effects.echo
```

This loads the submodule `sound.effects.echo`. It must be referenced with its full name.

```
sound.effects.echo.echofilter(input, output, delay=0.7, atten=4)
```

An alternative way of importing the submodule is:

```
from sound.effects import echo
```

This also loads the submodule `echo`, and makes it available without its package prefix, so it can be used as follows:

```
echo.echofilter(input, output, delay=0.7, atten=4)
```

Yet another variation is to import the desired function or variable directly:

```
from sound.effects.echo import echofilter
```

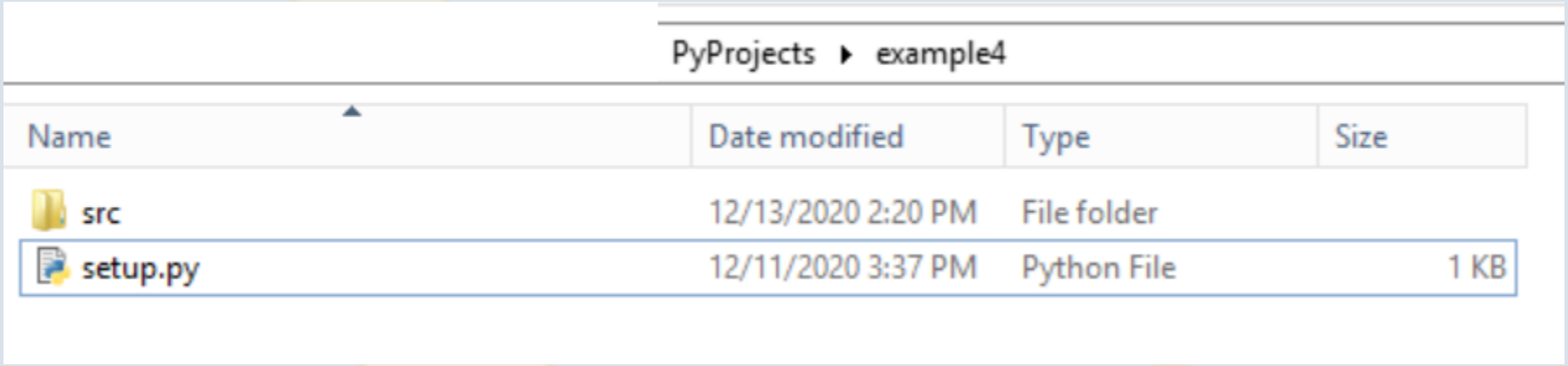

Python
Publication

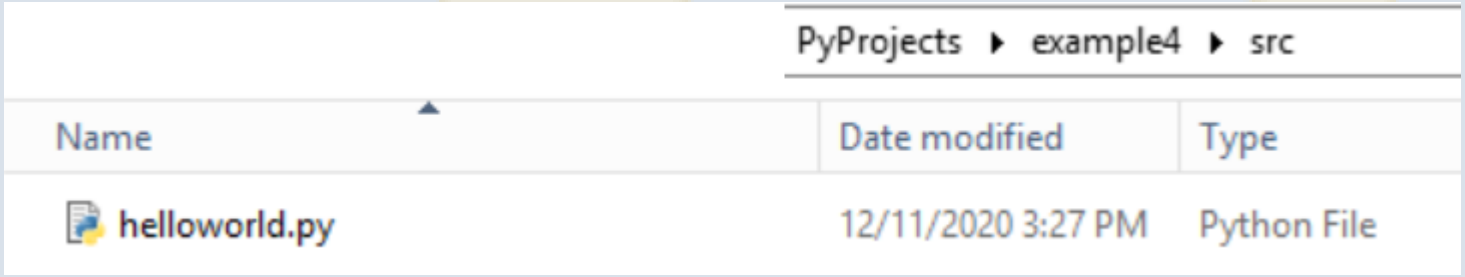
General

- Pack a package for distribution
- Either to collogues at work or to entire world (PyPi)
- Easy transfer of implementation
- Both source and binary code are supported

Preparation

- Place the source code under directory src
- Write setup.py (the publishing configuration)





Setup.py

- The minimum information is shown below
- setup() serves as a configuration – one parameter for each configuration item

```
from setuptools import setup

setup(
    name='helloworld',
    version='0.0.1',
    description='say hello!',
    py_modules=["helloworld"],
    package_dir={'': 'src'},
)
```

the name for pip install

The modules (files) to publish

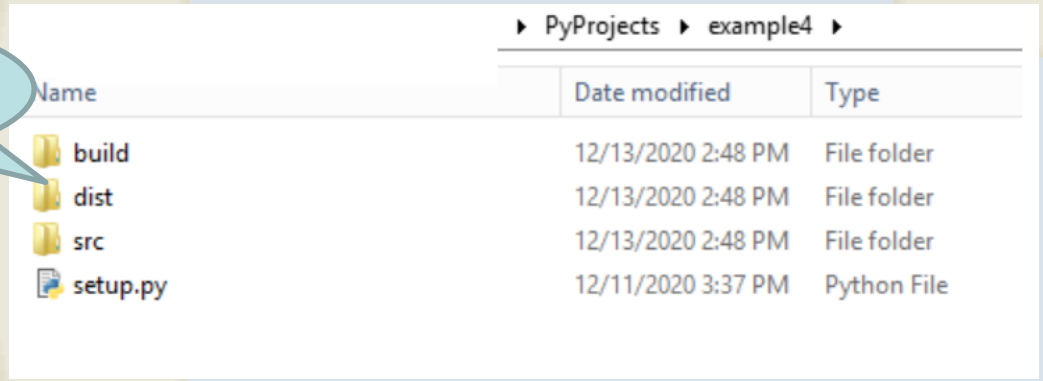
The directory to publish

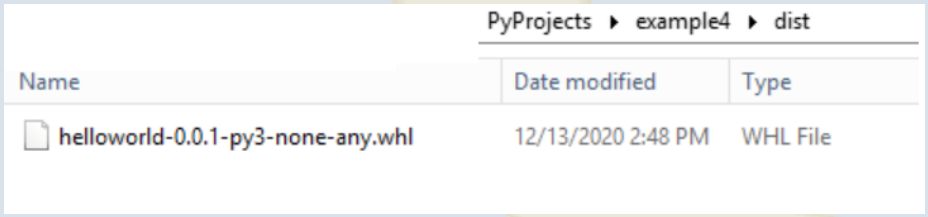
Executing

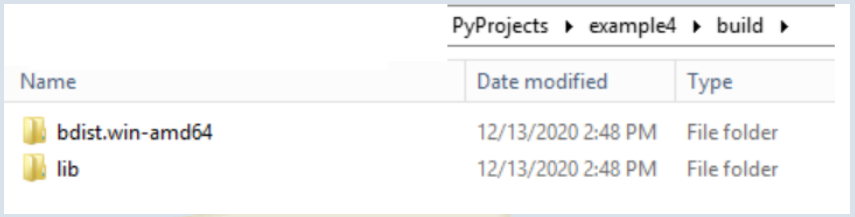
- Create the distribution data

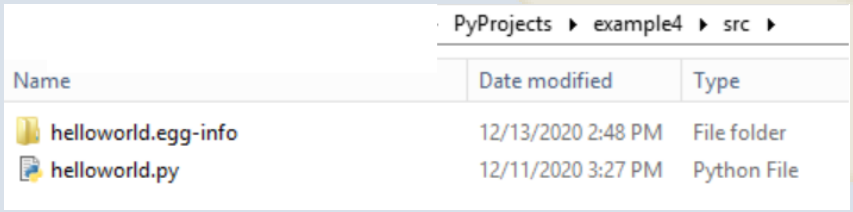
```
$ python setup.py bdist_wheel
```

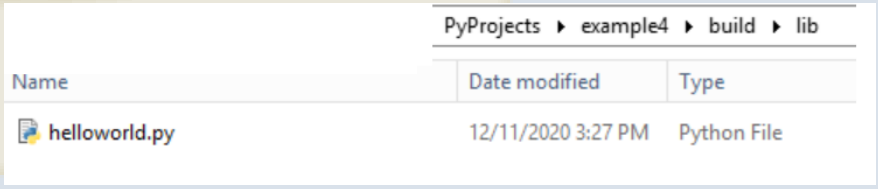
New directories build and dist











Installing

- Copying the code to a location that is reachable with the Python path

```
$ pip install -e .
```

- Now the code can be used with import everywhere in the same computer

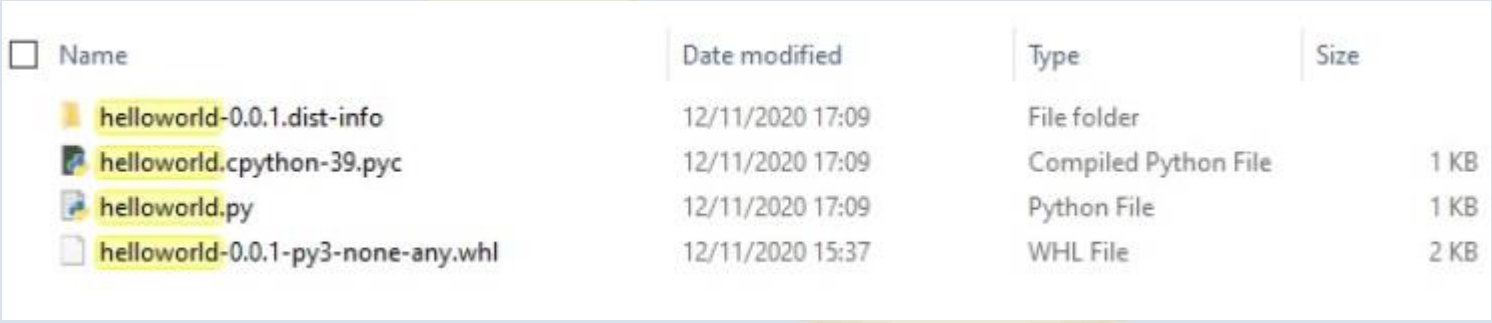
```
$ python

>>> from helloworld import say_hello

>>> say_hello()
'Hello, World!'
```

Installing

- Installing on another computer

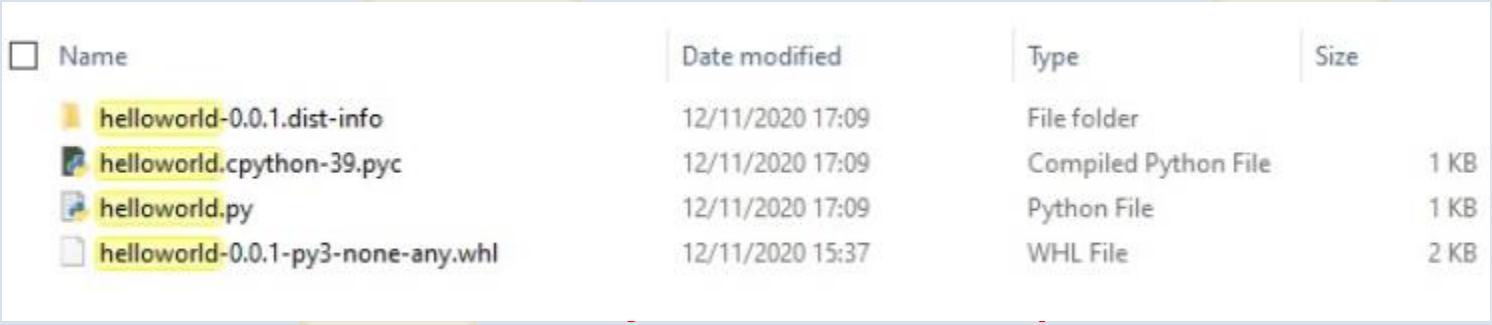
```
C:\DATA\python_test>pip install helloworld-0.0.1-py3-none-any.whl
Defaulting to user installation because normal site-packages is not writeable
Processing c:\data\python_test\helloworld-0.0.1-py3-none-any.whl
Installing collected packages: helloworld
Successfully installed helloworld-0.0.1

C:\DATA\python_test>python
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from helloworld import say_hello
>>> say_hello()
'Hello, world'
>>>
```

- The necessary files are there to be used



Installing

- The necessary files are there to be used



Name	Date modified	Type	Size
helloworld-0.0.1.dist-info	12/11/2020 17:09	File folder	
helloworld.cpython-39.pyc	12/11/2020 17:09	Compiled Python File	1 KB
helloworld.py	12/11/2020 17:09	Python File	1 KB
helloworld-0.0.1-py3-none-any.whl	12/11/2020 15:37	WHL File	2 KB